INGENIERÍA DE SEGURIDAD US
E.T.S. Ingeniería Informática. Universidad de Sevilla
Avda Reina Mercedes s/n. 41012 Sevilla
Tlf/Fax 954 557 139 **E-mail** lsi@lsi.us.es **Web**
www.lsi.us.es

PAI-3. BYODSEC-BRING YOUR OWN DEVICE SEGURO PARA UNA UNIVERSIDAD PÚBLICA USANDO ROAD WARRIOR VPN SSL

INTRODUCCIÓN

Los Road Warriors son usuarios remotos que necesitan acceso Seguro a las infraestructuras de sus organizaciones. En este proyecto, una determinada Universidad Pública nos solicita la implementación de la Política de Seguridad **Bring your Own Device (BYOD)** seguro para sus empleados, que consiste en que éstos utilicen sus propios dispositivos para realizar sus trabajos, pudiendo tener acceso a recursos de la Universidad tales como correos electrónicos, bases de datos y archivos en servidores corporativos usando una VPN SSL. Para la transmisión de todos estos elementos es fundamental la implementación de canales de comunicación seguros. En proyectos anteriores hemos visto cómo **proceder al aseguramiento de la información de las empresas y organizaciones** mediante tecnologías de **aseguramiento de la integridad usando mecanismo de HMAC.** Estos elementos además de otros muchos son fundamentales para la implementación de canales de comunicación seguros. Como se ha visto, una de las maneras para crear canales de comunicación seguros es la utilización de ***VPN (Virtual Private Networks)*** que usan en su implementación Sockets o bien implementaciones de terceros como ***OpenVPN***. Un Socket no es más que la especificación de una dirección IP y un puerto. La información puede transmitirse **en "claro" o cifrada** dependiendo del tipo de ***Socket*** que se utilice o de los algoritmos que se utilicen con la información a transmitir.

La política de seguridad de la organización nos dice con respecto a la seguridad en las transmisiones de información cliente-servidor:

*"… deberían ser **confidenciales e íntegras** y **además autenticadas**."*

Ya se han visto en proyectos anteriores varias formas para hacer la información integra, pero no confidencial y autenticada a través del canal de comunicaciones. Por ello en este proyecto una de las opciones posibles de implementación serían los sockets del tipo ***Secure Sockets Layers (SSL), que se han implementado en el lenguaje Java*** como se observa en la Figura 1.

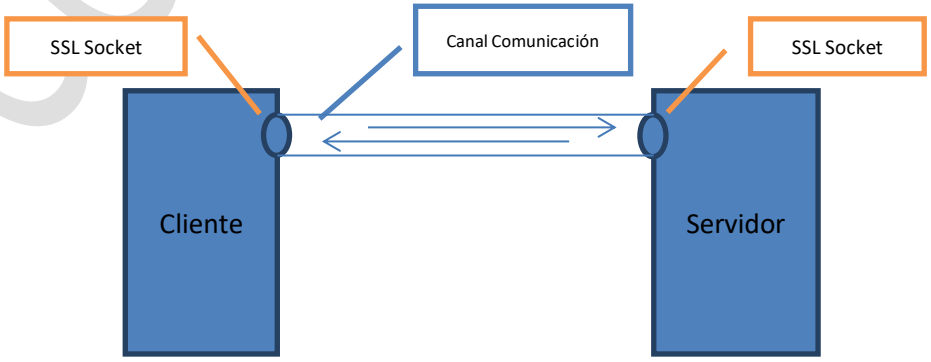

Figura 1: Canal de Comunicación Segura con SSL

SSL/TLS es un protocolo de comunicación seguro (implementa los requisitos de **autenticidad, confidencialidad e integridad**). Este protocolo utiliza una infraestructura basada en almacenes de claves y certificados.

Además, la Universidad Pública como Administración pública debe responder al cumplimiento legal exigido por la legislación española actual, al encontrarse la sede de dicha Universidad en España. En este caso, además del Reglamento de Protección de Datos Personales de la Unión Europea y de Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, se exige el cumplimiento del Real Decreto 3/2010 del Gobierno de España, de 8 de enero, por el que se regula el Esquema Nacional de Seguridad (ENS) en el ámbito de la Administración Electrónica. Dicho ENS indica en su artículo 1 que está "… ***constituido por los principios básicos y requisitos mínimos requeridos para una protección adecuada de la información. Será aplicado por las Administraciones públicas para asegurar el acceso, integridad, disponibilidad, autenticidad, confidencialidad, trazabilidad y conservación de los datos, informaciones y servicios utilizados en medios electrónicos que gestionen en el ejercicio de sus competencias.***

Lo exigido en esta Administración Pública se encuentra bien soportado por la ***Open Security Architecture (OSA)*** donde se define el patrón de seguridad ***SP-008: Public Web Server Pattern*** (aunque en este caso no es un servidor Web***)***, donde para una infraestructura con un servidor público, se describe el control ***SC-09 Transmision confidentiality:***

*" **Control**: The information system protects the confidentiality of transmitted information."*

Objetivos

1. Desarrollar/seleccionar **cómo llevar a la práctica de forma lo más eficiente posible los canales de comunicación segura para la transmisión *de credenciales (usuario, contraseñas) y un mensaje*** con el Protocolo SSL/TLS (autenticidad, confidencialidad e integridad). Tener en cuenta que **el número de empleados que usarán la aplicación son aproximadamente 300.**
2. Utilizar alguna herramienta de análisis de tráfico que **permita comprobar la confidencialidad e integridad de los canales de comunicaciones seguros**.
3. Establecer los ***Cipher Suites*** que serán usados en la versión TLS 1.3. Además, **el cliente nos solicita pruebas sobre la capacidad para soportar a los 300 empleados por la VPN SSL desarrollada**.

   Nota: Es muy importante para el **cliente que la implementación de la VPN que se implemente sea los más eficiente posible**. Considerando eficiente aquella solución con bajo *overhead*, baja utilización del *ancho de banda*, baja *latencia* y alta *configurabilidad* y *escalabilidad*.

Tareas para desarrollar usando una infraestructura Java

1. Arquitectura Cliente-Servidor Segura

En primer lugar, vamos a crear una infraestructura cliente-servidor para la **comprobación de login/password** de usuarios y un mensaje **secreto** mediante el uso de sockets seguros.

Se puede realizar el diseño de dicha arquitectura de diferentes maneras que cada Security Team debería decidir, en función de **la eficacia y eficiencia**. Aquí mostramos un ejemplo específico usando la infraestructura en el lenguaje Java, para lo que seguiremos los siguientes pasos:

A. Creación de un keyStore – Compartición de claves

De acuerdo con la especificación del protocolo SSL/TLS se necesita para la autenticación de los servidores de los correspondientes certificados. En Java se puede realizar esto mediante la creación

de un almacén o repositorio de certificados de seguridad para el protocolo SSL. Java viene provisto de una herramienta que permite crear de manera sencilla un almacén de certificados. Para crear el almacén de certificados ejecutamos en consola y con permisos de administración el siguiente comando:

```
Keytool -genkey -keystore C:\SSLStore\...\keystore.jks -alias SSLCertificate -keyalg
RSA
```

Keytool hará una serie de preguntas para crear la creación del keyStore. Con esto ya tendremos en la ruta C:\ un archivo keystore.jks el almacén de certificados que usaran nuestros Sockets cliente y servidor. De manera similar se podría realizar en un sistema basado en Linux.

Nota: Para usuarios de Windows, ***keytool*** se encuentra en la carpeta bin del JDK instalado en el sistema. Por lo tanto, para usar ***keytool*** desde línea de comandos sin tener que entrar en la ruta debemos configurar las variables de entorno JAVA_HOME y PATH con la ruta hacia la carpeta bin de la JDK de Java.

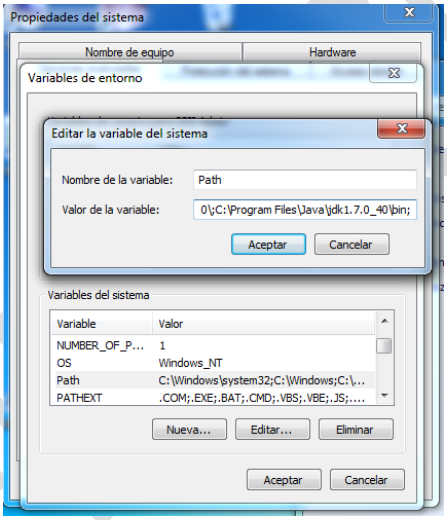

Figura 2: Configuración variable de entorno

Observe toda la información que se le pide al crear el keyStore **¿Para qué cree que puede ser necesario esta información?** Recoja todas sus reflexiones y el trabajo realizado para el informe y en su presentación.

B. Generación de un Socket TLS Servidor

Para utilizar comunicaciones seguras usando SSL/TLS entre cliente-servidor, utilizaremos Socket SSL/TLS. Crearemos una clase ***BYODServer.java*** con el mismo código que la clase LoginServer.java de la PAI-2 pero creando un SSL Socket Server en lugar de un Server Socket:

```
SSLServerSocketFactory socketFactory = (SSLServerSocketFactory)
                                        SSLServerSocketFactory.getDefault();
serverSocket = (SSLServerSocket) socketFactory.createServerSocket(7070);
```

El código se puede ejecutar en Eclipse, pero debemos tener en cuenta que son dos procesos (servidor y cliente) diferentes con diferentes parámetros, por tanto, deberemos tener activada la correspondiente configuración de Eclipse. O también podemos no ejecutar directamente este código sobre Eclipse, y hacerlo en el entorno de ejecución del sistema operativo especificando los procesos y los parámetros a la JVM de dónde se encuentra el almacén de claves y la clave de este como veremos en el apartado D.

C. Generación de un Socket TLS Cliente

Crearemos una clase ***BYODCliente.java*** con la única diferencia con respecto al código de la PAI-2 en la creación de los **Sockets**, en este caso utilizamos las factorías ***SSLSocketFactory*** tal y como se muestra a continuación:

```
SSLSocketFactory socketFactory = (SSLSocketFactory)  SSLSocketFactory.getDefault();
SSLSocket socket = (SSLSocket) socketFactory.createSocket("localhost", 7070);
```

Notar que la IP y el puerto de conexión deben coincidir con el puerto e IP del servidor.

D. Compilar y ejecutar código cliente y servidor desde consola

1. Desde una consola con permisos de administración y en el directorio donde se encuentre la clase Java del cliente o servidor, ejecutaremos el siguiente comando:

   ```
   javac BYODServer.java
   ```

   Esto va a generar un archivo .class con el código interpretable por la JVM de Java, y que utilizaremos desde la misma línea de comandos para su ejecución.

2. Para ejecutar las clases una vez compiladas simplemente debemos de ejecutar el siguiente comando donde:

   - **BYODServer:**

     ```
     java "-Djavax.net.ssl.keyStore=C:\SSLStore\...\keystore.jks" "-
     Djavax.net.ssl.keyStorePassword=PASSWORD" BYODServer
     ```

   - **BYODCliente:**

     ```
     java "-Djavax.net.ssl.trustStore=C:\SSLStore\...\keystore.jks" "-
     Djavax.net.ssl.trustStorePassword=PASSWORD" BYODCliente
     ```

3. Compruebe que, si un empleado escribe en la correspondiente interfaz su **username**, su **contraseña**, y un **mensaje**, estos se envían sin problemas de seguridad y el servidor tras comprobar que es un usuario y password correcto, le informa al cliente que su mensaje SECRETO ha sido almacenado correctamente, en caso contrario le indicará que su mensaje no se ha almacenado.
4. **La versión TLS que nos ha requerido la entidad pública para las comunicaciones es TLS 1.3, por tanto, no se debe usar versiones anteriores.**

2. Análisis de tráfico de red en comunicaciones (sniffers)

Una manera de inspeccionar la información que fluye entre los Sockets es usar herramientas para analizar tráfico de red o sniffers. Los sniffers se interponen en los canales de comunicación y capturan toda la información que fluye por ellos. Existen sniffers muy sofisticados como *Wireshark* que permiten analizar múltiples protocolos, y otros sniffers más simples, por ejemplo, tcpdump que sólo capturan la información de red y hacen un volcado de la información de los paquetes en un archivo dump.

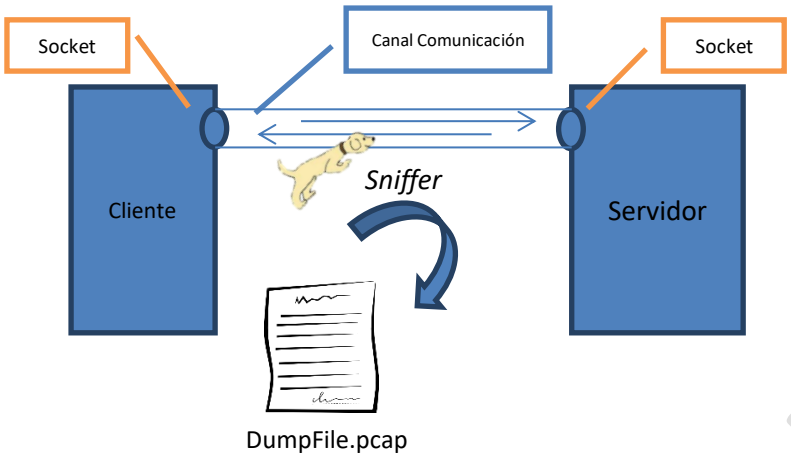

Figura 3: Monitorización del tráfico de red

En este proyecto si estamos en Windows, podríamos usar *RawCap* un sniffer para Windows que se utiliza desde línea de comandos, y/o *Wireshark. P*ara inspeccionar el archivo de salida de RawCap podremos usar Wireshark. En Linux no es necesario usar *RawCap*, podría usarse directamente Wireshark o un sniffer como *tcpdump* junto con *Wireshark.*

RawCap

RawCap nos hará dos preguntas: (1) la interfaz queremos escuchar; y (2) ruta y nombre del archivo donde queremos volcar la información (con extensión pcap). En nuestro caso vamos a capturar el tráfico en *loopback* o *localhost*, ya que el canal de comunicación se estable en nuestra propia máquina. *RawCap* interceptará todos los paquetes de información y los escribirá en el archivo especificado. En la siguiente imagen vemos a *RawCap* en marcha escuchando en la interfaz loopback, y la ruta (c:\ dumpfile.pcap) del fichero donde escribirá la información.

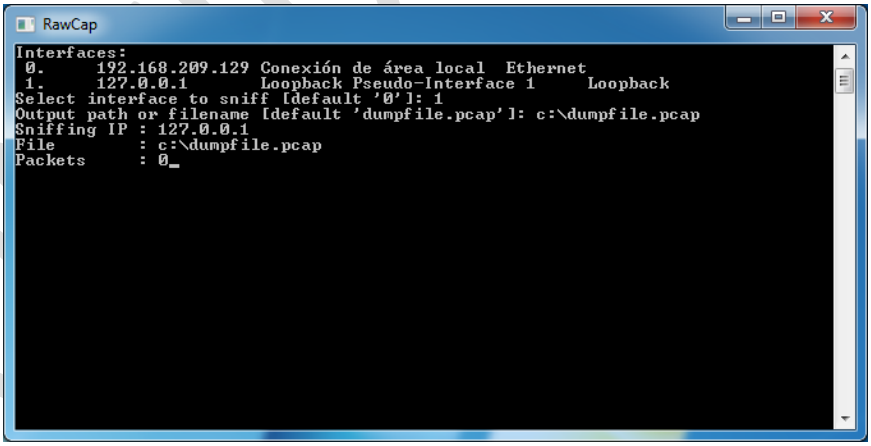

Figura 4: RawCap capturando información en la interfaz localhost

Wireshark

Una vez capturado el tráfico con ***RawCap,*** y obtenido el archivo .pcap se puede abrir con la herramienta ***Wireshark*** de tal manera que podemos observar el intercambio de paquetes que se ha producido en la interfaz seleccionada.

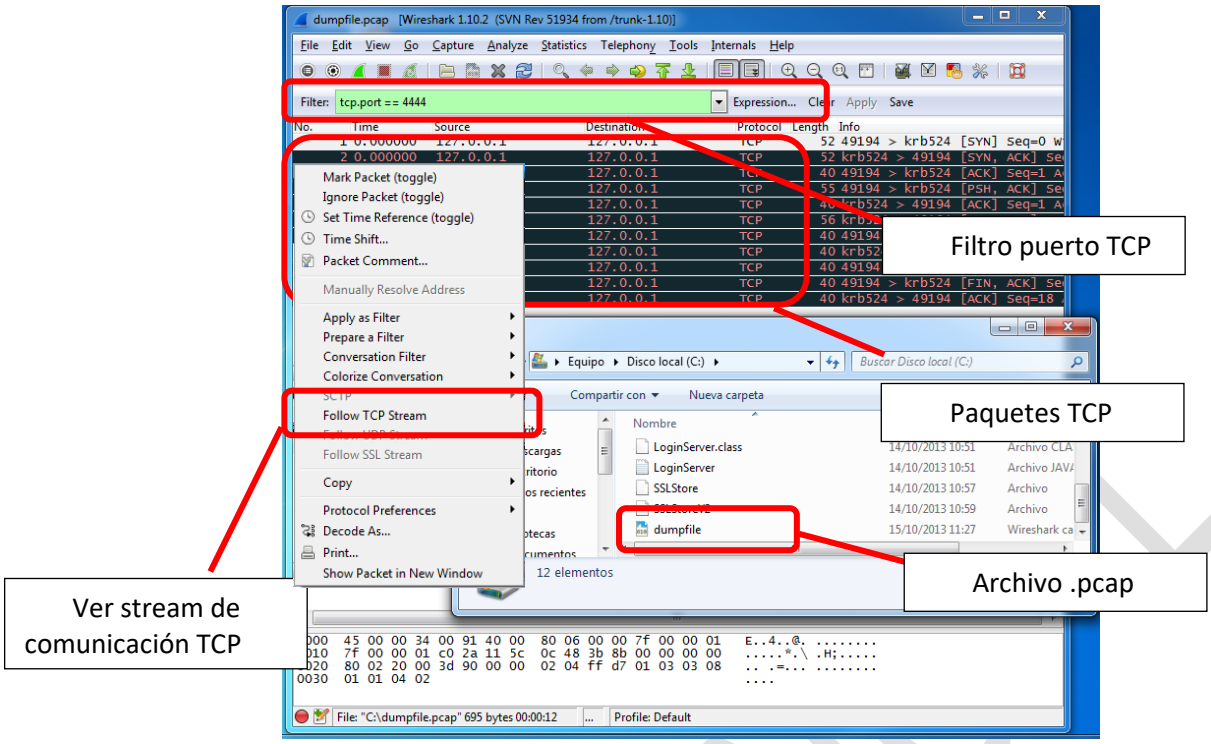

Figura 5: Configuración de Wireshark

El sniffer ha podido capturar múltiples paquetes que pasan por la interfaz loopback, en nuestro caso vamos a filtrar los paquetes capturados para observar sólo aquellos que utilicen el puerto 4444 como se observa en la Figura 2. Si utilizamos la opción de *"Follow TCP Stream"* podemos observar todos los datos que se transmiten en la conexión de los Sockets que hemos llevado a cabo.

Haga uso de un analizador de tráfico de red (podría ser Wireshark, RawCap para Windows, ...) para observar la información en el canal de comunicación. **¿Podemos obtener los datos de nombre de usuario, contraseña o el mensaje secreto?** Además, haga una reflexión acerca de que está ocurriendo con el keyStore **¿se está utilizando el mismo keyStore? ¿Qué ocurriría si utilizamos dos máquinas distintas, donde estaría el keyStore?** Muestre en su informe y en la presentación todas las conclusiones acerca de esta tarea.

3. Selección e implementación del conjunto de cipher suite más adecuado para la seguridad con SSL/TLS

Uno de los aspectos más relevantes para disponer de un **nivel de seguridad "fuerte" (strong en inglés)** en las comunicaciones SSL/TLS sería poder escoger la correspondiente ***Cipher Suite*** por parte de los clientes para realizar las comunicaciones seguras con el servidor. Se pretende entonces **descubrir a través del tráfico generado en las comunicaciones la *Cipher Suite* que ha utilizado** por defecto cuando ha usado los ***Socket SSL*** en la última tarea.
Presente en el informe el trabajo realizado para descubrir ello y analice el grado de seguridad que tiene esta ***Cipher Suite y* cómo la podríamos cambiar si lo consideramos poco segura e implemente dicho cambio en la solución**.

Normas del entregable

- Cada grupo debe entregar a través de la Plataforma de Enseñanza Virtual y en la **actividad** preparada para ello un archivo zip, nombrado **PAI3-ST_*Num*.zip**, que deberá contener al menos los ficheros siguientes:
  - ✓ **Documento en formato pdf que contenga un informe/resumen del proyecto** con los detalles más importantes de las decisiones, soluciones adoptadas y/o

implementaciones desarrolladas, así como el resultado y análisis de las pruebas realizadas (máximo 10 páginas).

- **Código fuente de las posibles implementaciones y/o scripts desarrollados y/o configuraciones y/o logs del analizador de código.**
- El plazo de entrega de dicho proyecto finaliza el **día 5 de abril a las 10:30 horas.**
- Los proyectos entregados fuera del plazo establecidos serán considerados inadecuados por el cliente y por tanto entrarán en penalización por cada día de retraso entrega de 5% del total, hasta agotarse los puntos.
- **El cliente no se aceptará envíos realizados por email, ni mensajes internos de la enseñanza virtual, ni correo interno de la enseñanza virtual. Toda entrega realizada por estos medios conllevará una penalización en la entrega del 5%.**

Métricas de valoración

Para facilitar el desarrollo de los equipos de trabajo el cliente ha decidido listar las métricas que se tendrán en cuenta para valorar los entregables de cada grupo de trabajo:

- **Resumen (30%)**
  - Tamaño del informe
  - Calidad del resumen aportado
  - Calidad de pruebas presentadas y resultados
- **Solución aportada (70%)**
  - Cumplimiento de requisitos establecidos
  - Calidad del código entregado
  - Complejidad de la solución
  - **Eficiencia de la solución**
  - Respuesta al conjunto de preguntas planteadas
  - Pruebas realizadas
- **Extras de Productividad (20%):**
  - Aquellos Security Team que puedan mostrar **en las sesiones de seguimiento del PAI** el funcionamiento de la comunicación cliente/servidor sin estar ambos en la misma máquina **se elevará por INSEGUS el grado de satisfacción de dicho PAI un 10%.**
  - Aquellos Security Team que muestren **en las sesiones de seguimiento del PAI** que el análisis del tráfico de red se realiza usando una tercera máquina (ejemplo: podría actuar como un proxy o se conecta a través de alguna roseta de conexión de red**) se elevará por INSEGUS otro 10% el grado de satisfacción de dicho PAI.**